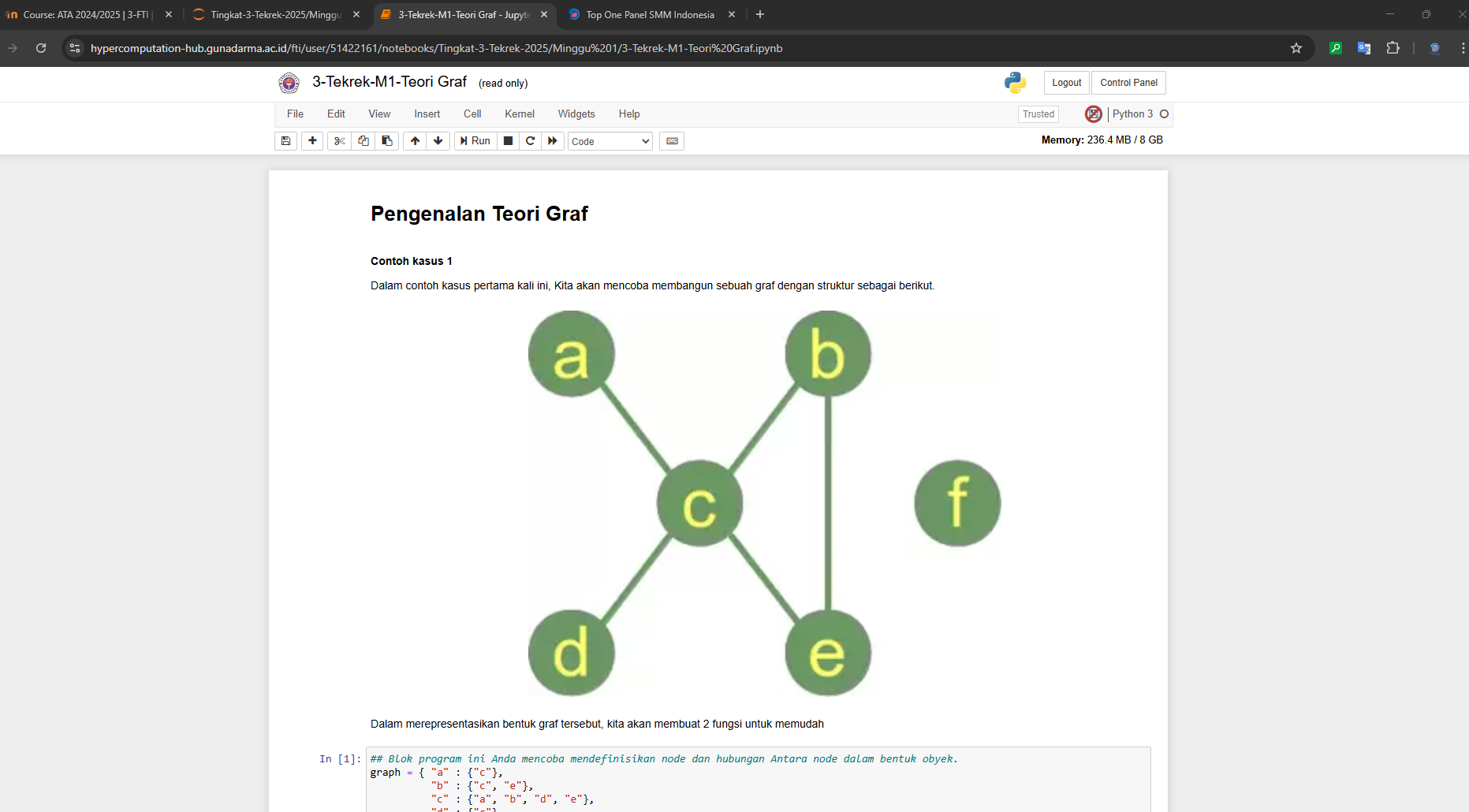
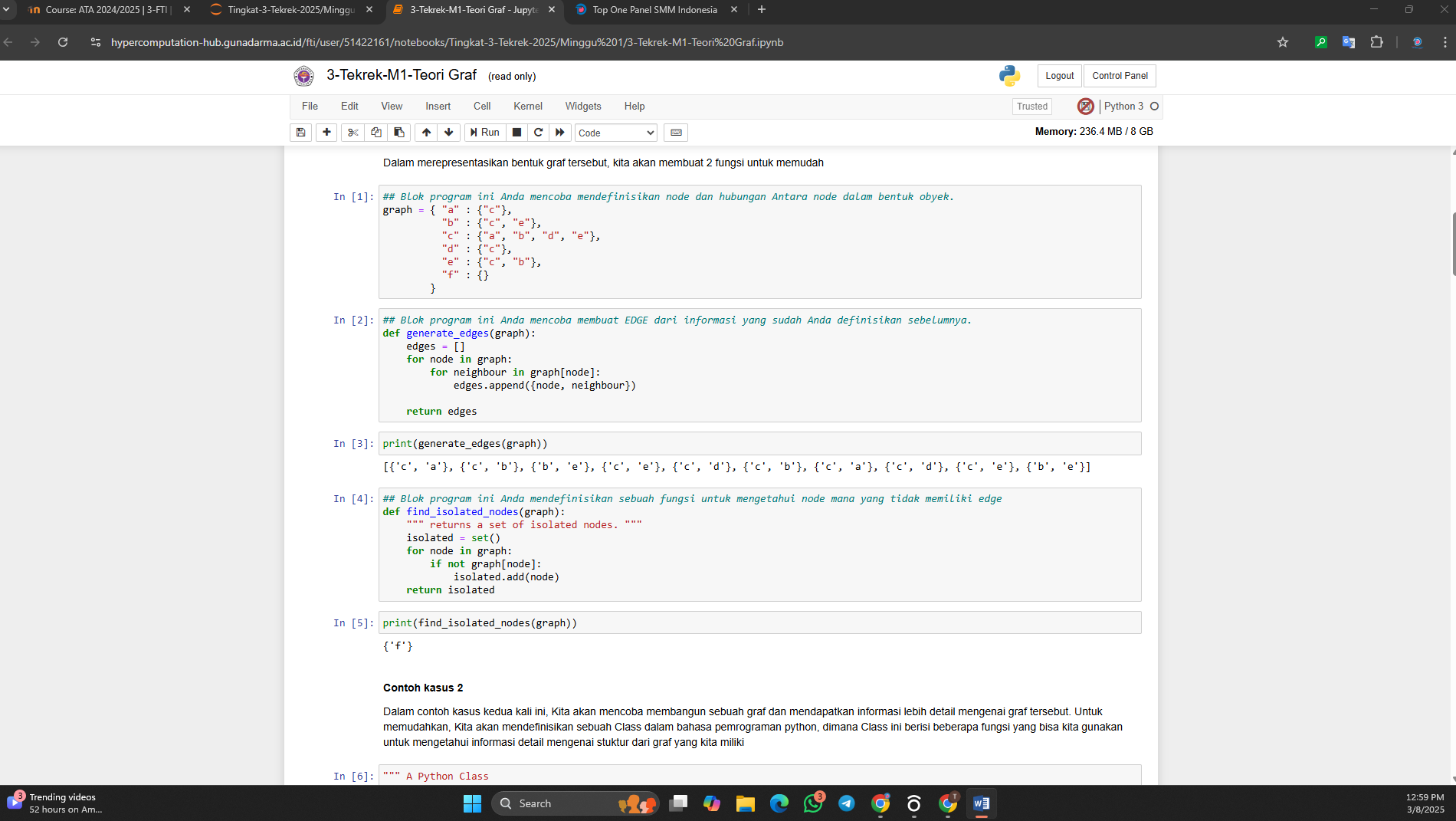
MUHAMMAD TARMIDZI BARIQ

51422161

3IA11





1

graph = { "a" : {"c"},

"b" : {"c", "e"},

"c" : {"a", "b", "d", "e"},

"d" : {"c"},

"e" : {"c", "b"},

"f" : {}

}

graph adalah variable

node/simpul dalam graf (yaitu "a", "b", "c", "d", "e", dan "f")

2

def generate\_edges(graph): ## Mendefinisikan fungsi bernama generate\_edges yang menerima parameter graph

edges = [] ## Membuat list kosong bernama edges yang akan digunakan untuk menyimpan

for node in graph: Memulai iterasi/perulangan untuk setiap node (simpul) dalam dictionary graph

for neighbour in graph[node]: ## Untuk setiap node, melakukan iterasi terhadap semua tetangga (neighbour) dari node tersebut

edges.append({node, neighbour}) ## Menambahkan edge berupa set {node, neighbour} ke dalam list edges

return edges ## Setelah semua iterasi selesai, fungsi mengembalikan list edges yang berisi semua edge dalam graf.

3

print(generate\_edges(graph)) ## melakukan print dari fungsi generate\_edges dengan parameter graph

4

def find\_isolated\_nodes(graph): ## Mendefinisikan fungsi bernama find\_isolated\_nodes yang menerima parameter graph

""" returns a set of isolated nodes. """ ## fungsi ini mengembalikan set berisi node-node yang terisolasi.

isolated = set() ## Membuat set kosong bernama isolated

for node in graph: ## Memulai iterasi/perulangan untuk setiap node (simpul) dalam dictionary graph.

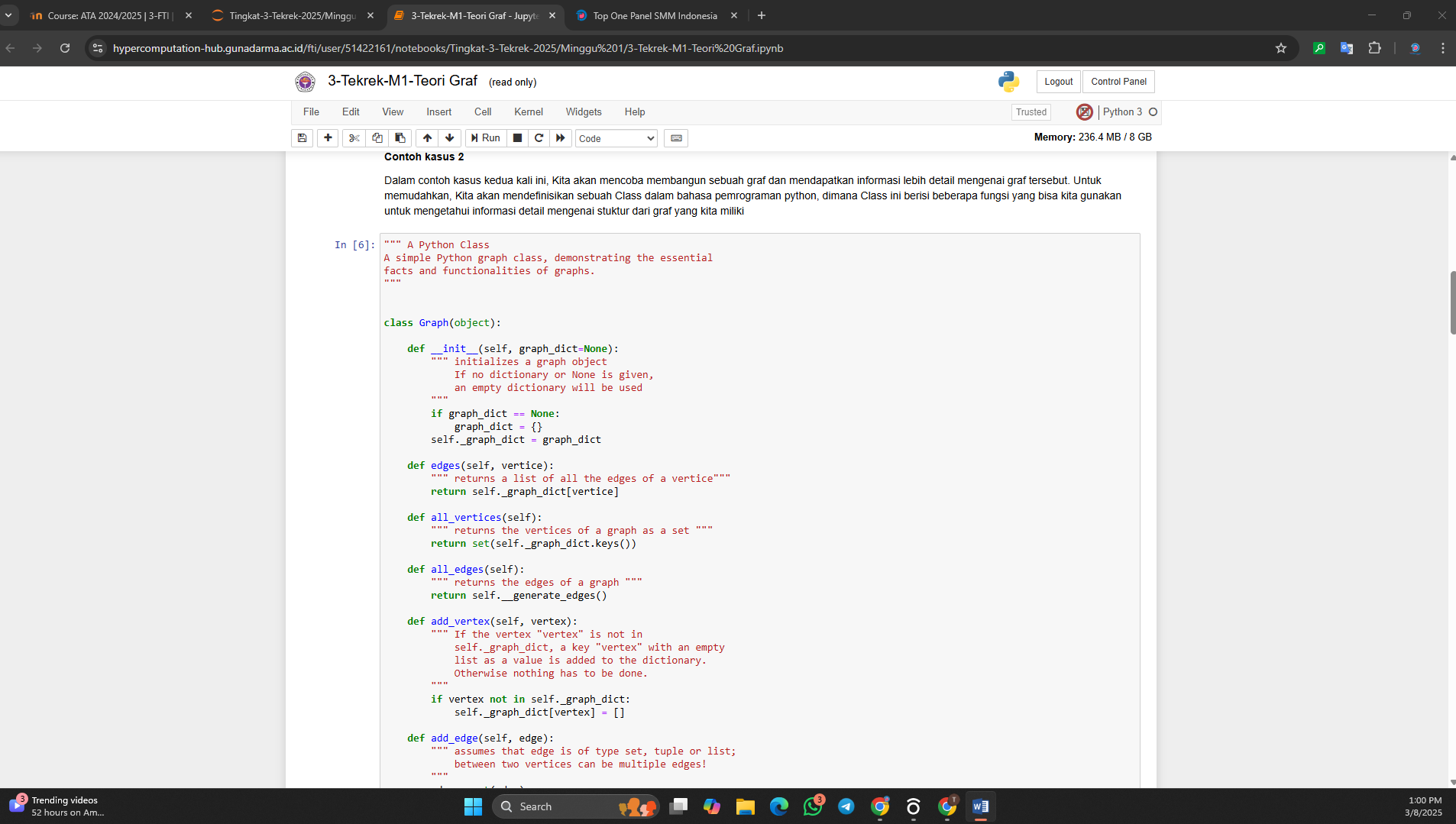
if not graph[node]: ## Memeriksa apakah nilai (value) yang terkait dengan node saat ini adalah kosong. Jika graph[node] adalah set kosong, maka kondisi ini bernilai True.

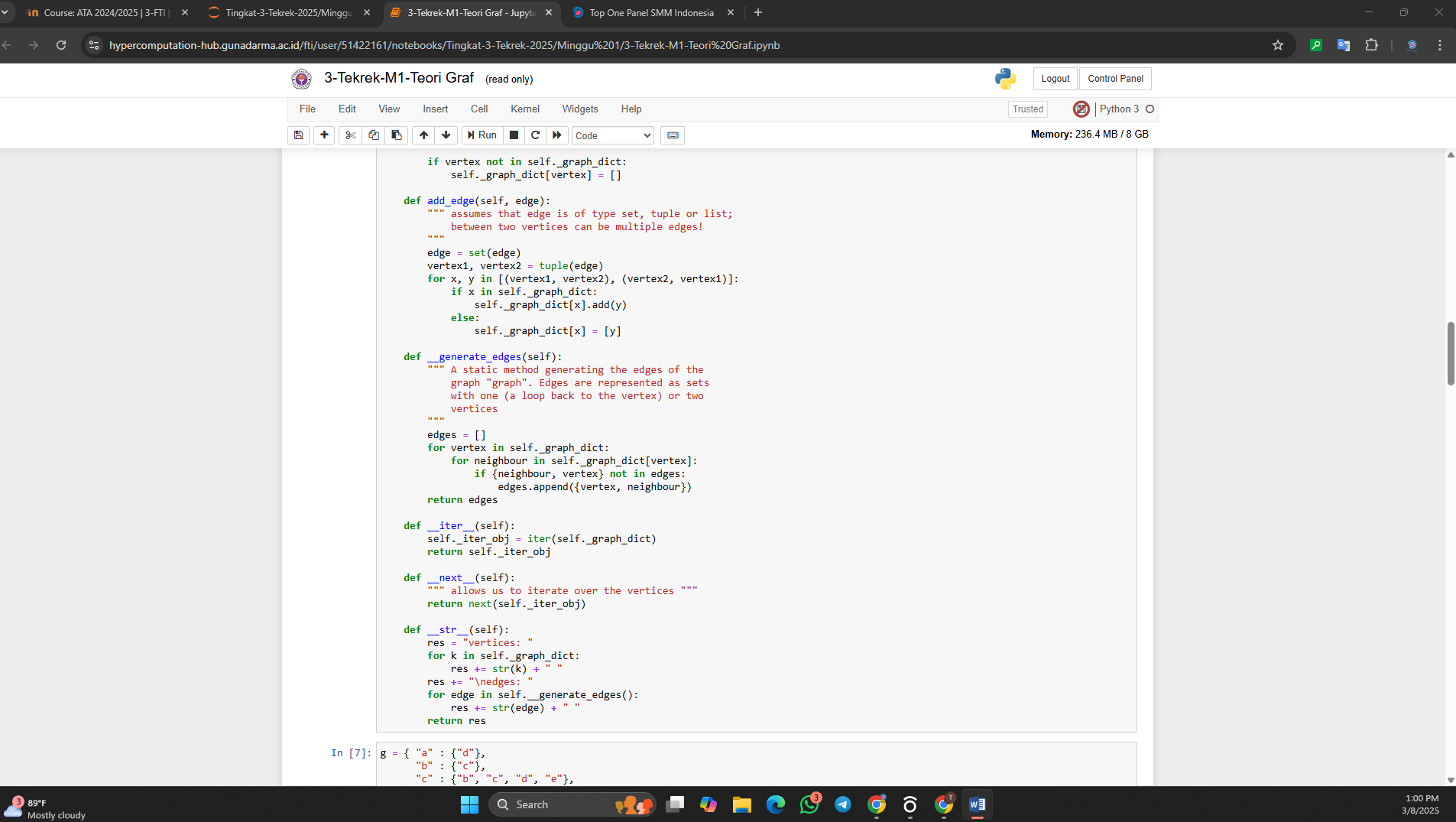
isolated.add(node) ## Jika node tidak memiliki tetangga (nilai set kosong), tambahkan node tersebut ke dalam set isolated menggunakan metode add().

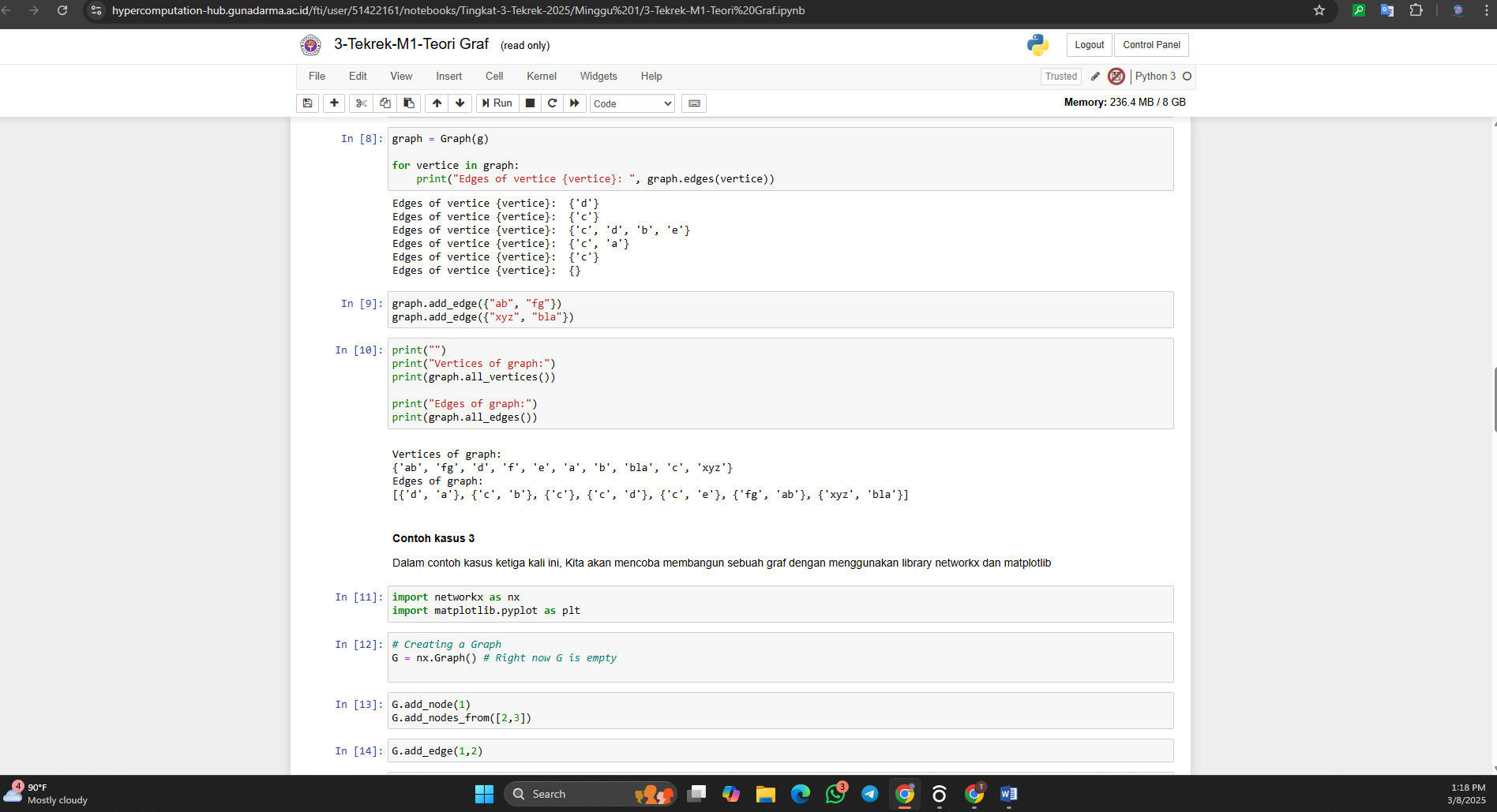
return isolated ## Setelah semua iterasi selesai, fungsi mengembalikan set isolated yang berisi semua node terisolasi dalam graf.

5

print(find\_isolated\_nodes(graph)) ## melakukan print dari fungsi find\_isolated\_nodes dengan parameter graph







1

g = { "a" : {"d"},

"b" : {"c"},

"c" : {"b", "c", "d", "e"},

"d" : {"a", "c"},

"e" : {"c"},

"f" : {}

}

graph adalah variable

node/simpul dalam graf (yaitu "a", "b", "c", "d", "e", dan "f")

2

graph = Graph(g)

for vertice in graph:

print("Edges of vertice {vertice}: ", graph.edges(vertice))

Kode tersebut menggunakan objek Graph yang dibuat dari variabel g (yang tidak ditampilkan dalam kode), kemudian melakukan iterasi untuk setiap vertice (simpul) dalam graf dan mencetak semua edges (tepi) yang terhubung dengan vertice tersebut.

3

graph.add\_edge({"ab", "fg"})

graph.add\_edge({"xyz", "bla"})

menambahkan edge (tepi) baru ke dalam graf.

4

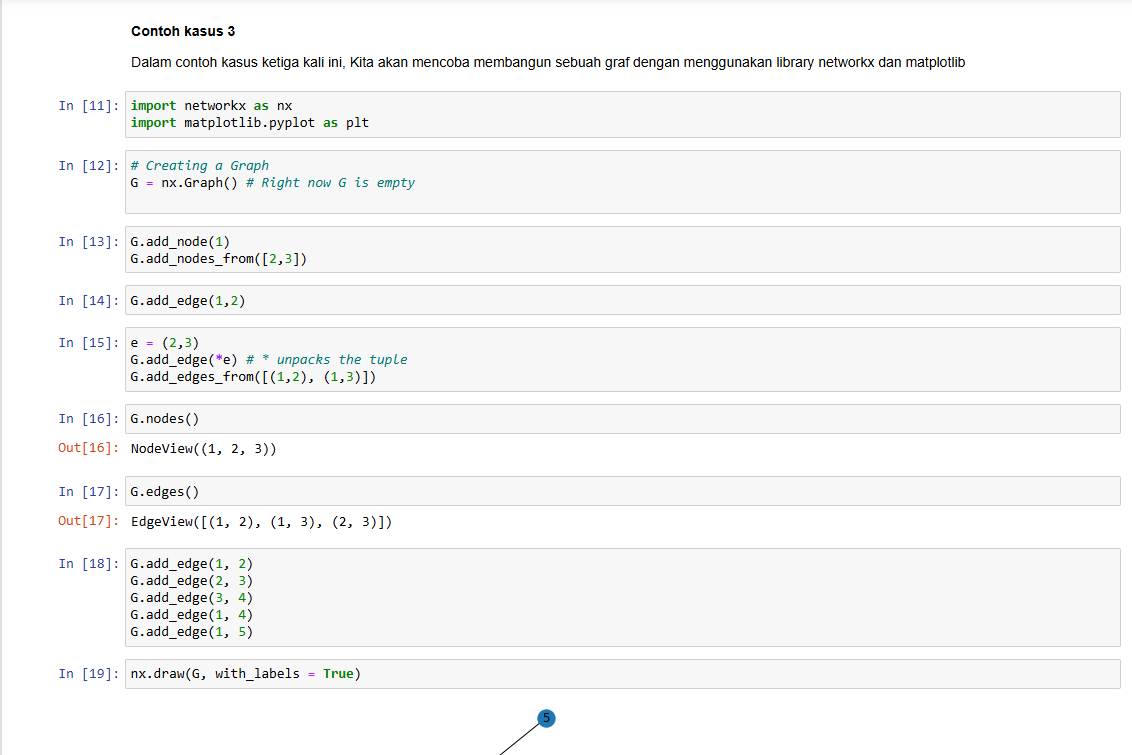
print("") ## Mencetak baris kosong sebagai pemisah.

print("Vertices of graph:") ## Mencetak judul/label untuk daftar simpul.

print(graph.all\_vertices()) ## Memanggil metode all\_vertices() dari objek graph dan mencetak hasilnya. Metode ini mengembalikan daftar semua simpul yang ada dalam graf.

print("Edges of graph:") ## Mencetak judul/label untuk daftar tepi.

print(graph.all\_edges()) ## Memanggil metode all\_edges() dari objek graph dan mencetak hasilnya. Metode ini mengembalikan daftar semua tepi yang ada dalam graf.



1

import networkx as nx ## Mengimpor library NetworkX yang disingkat sebagai 'nx'

import matplotlib.pyplot as plt ## Mengimpor submodule pyplot dari library Matplotlib dan menyingkatnya sebagai 'plt'

2

G = nx.Graph() ## membuat graph

3

G.add\_node(1) ## Menambahkan satu node dengan label 1 ke dalam graf.

G.add\_nodes\_from([2,3]) ## Menambahkan beberapa node sekaligus dari list [2,3], yaitu node 2 dan node 3, ke dalam graf.

4

G.add\_edge(1,2) ## menambahkan satu edge (tepi) ke dalam graf NetworkX yang bernama G.

5

e = (2,3) ## Membuat tuple yang berisi node 2 dan node 3.

G.add\_edge(\*e) ## Menambahkan edge antara node 2 dan node 3. Operator \* membuka (unpacks) tuple e sehingga nilainya dijadikan argumen terpisah, sama seperti menulis G.add\_edge(2,3).

G.add\_edges\_from([(1,2), (1,3)]) ## Menambahkan beberapa edge sekaligus dari list tuple: edge antara node 1 dan 2, serta edge antara node 1 dan 3.

6

G.nodes() ## metode pada objek graf NetworkX yang mengembalikan objek NodeView (tampilan node)

7

G.edges() ## metode pada objek graf NetworkX yang mengembalikan objek EdgeView (tampilan edge).

8

G.add\_edge(1, 2)

G.add\_edge(2, 3)

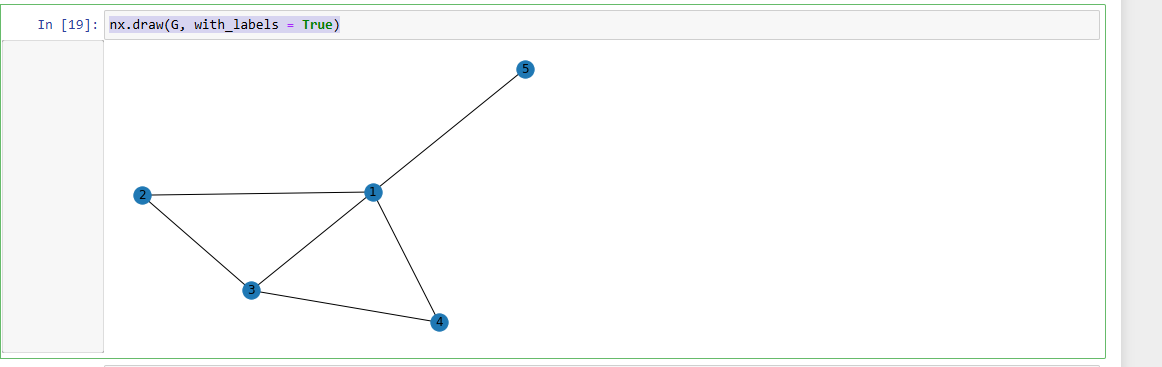
G.add\_edge(3, 4)

G.add\_edge(1, 4)

G.add\_edge(1, 5)

Kode ini menambahkan lima edge (tepi) ke dalam graf NetworkX yang bernama G

9

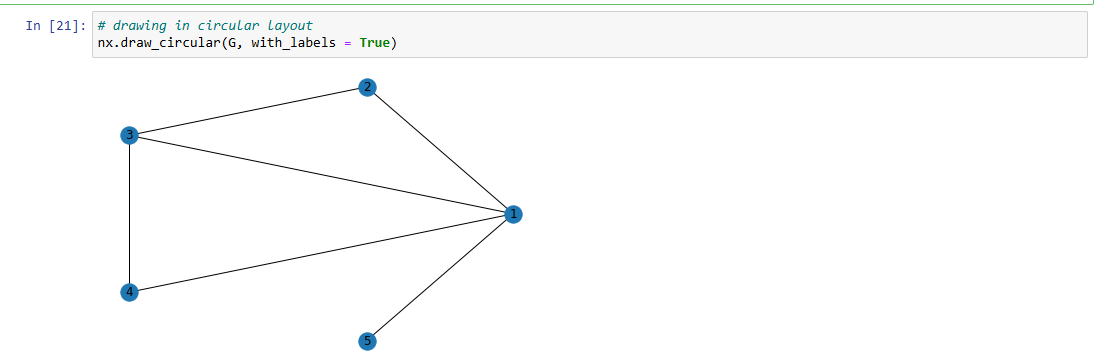


perintah untuk menggambar (visualisasi) graf NetworkX menggunakan Matplotlib.

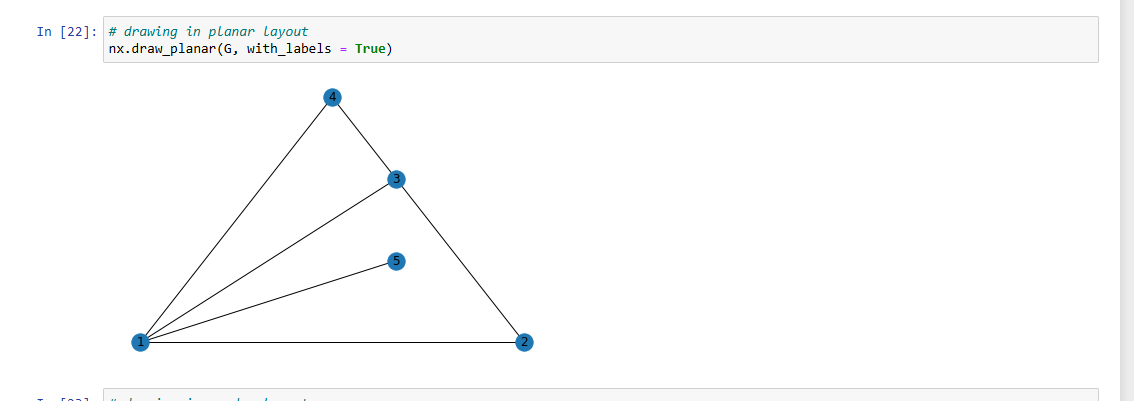
10

plt.savefig("contoh-graf-1.png") ## digunakan untuk menyimpan visualisasi graf yang telah dibuat sebelumnya sebagai file gambar.

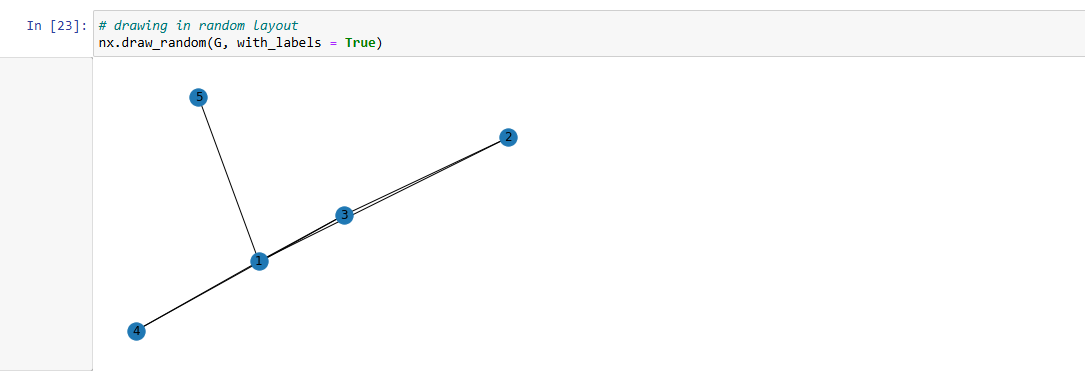
11



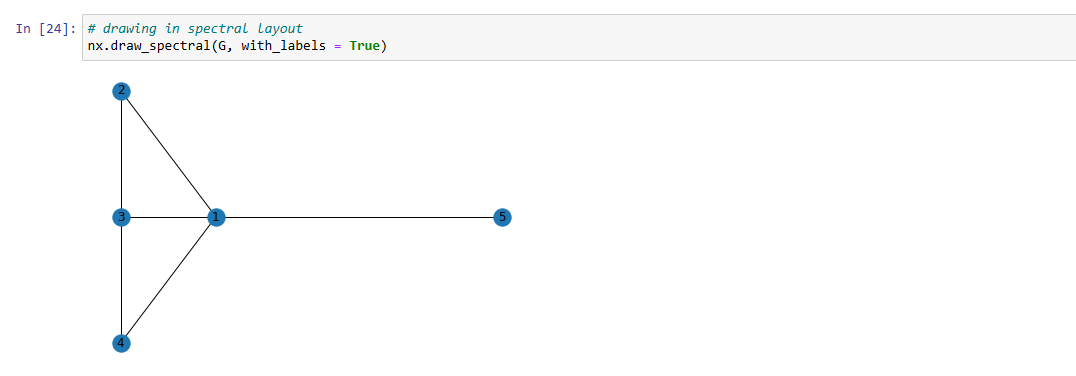
12



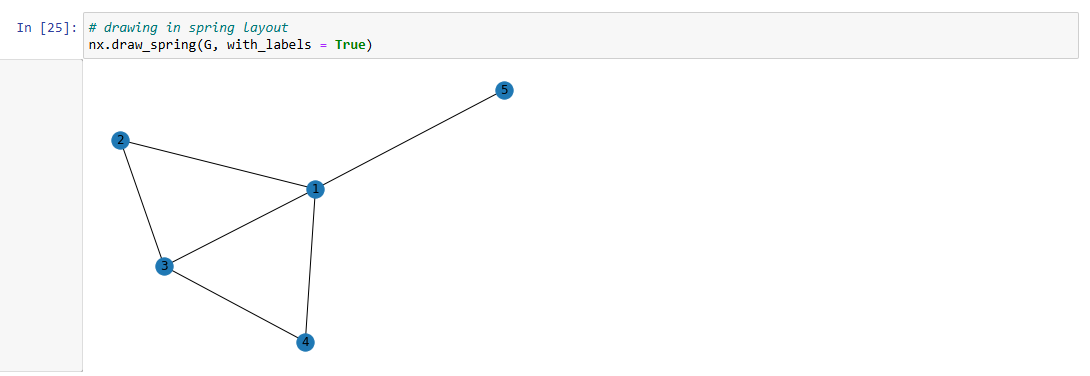
13



14



15



16

